Corso di Linguaggi di Programmazione
Prova scritta del 20 Giugno 2019.

Tempo a disposizione: ore 2.
Svolgere gli esercizi 1-4 e 5-8 su due fogli differenti.

1. Una strategia di valutazione *parallela* consente la valutazione in qualsiasi ordine degli argomenti di un'operatore. Considerando l'espressione booleana b_0 **or** b_1, fornire le regole SOS secondo la strategia EP (esterna-parallela). Mostrare un esempio di un'espressione di quel tipo per cui la valutazione EP e la valutazione ES (esterna-sinistra, vista a lezione) differiscono.

2. Se $L = \{a, ab\}$ e $R = \{b\}$, che linguaggio è $L \cdot R^*$? Tale linguaggio è descrivibile con una espressione regolare? In generale, se L e R sono regolari, il linguaggio $L \cdot R^*$ è regolare o libero, oppure non libero? Giustificare la risposta.

3. Costruire un parser LL(1) per il linguaggio $L = \{a^{2n}b^n \mid n \geq 0\}$. Mostrare il funzionamento del parser sull'input *aab*.

4. Si consideri la seguente grammatica G con simbolo iniziale S:

$$\begin{array}{lcl} S & \rightarrow & \mathtt{b}S\mathtt{c} \mid \mathtt{b}A\mathtt{c} \\ A & \rightarrow & \mathtt{a}A \mid \mathtt{a} \end{array}$$

   (i) Che linguaggio genera G? (ii) Si costruisca il parser SLR(1). (iii) Si mostri il funzionamento del parser su input *baac*.

5. Si consideri il seguente frammento in uno pseudolinguaggio con scope statico, parametri di ordine superiore, deep binding, passaggio per valore e per nome. Si dica cosa stampa il frammento motivando brevemente la risposta.

```
{void foo (int f(), int value n, int name x){
    int y = 10;
    void  fie(){
      write(n,x);
    }
    if (n==1) f();
        else    {x = 30;
                 foo(fie,1, x);
                 }
    }
   {
   int y = 100;
   void  g(){
        write(10);
        }
   foo(g,2,y);
   write(y);
   }
}
```

6. La definizione di certo linguaggio di programmazione specifica che la valutazione procede da sinistra a destra. Inoltre, nel valutare un'espressione complessa, eventuali sottoespressioni che vi compaiono più di una volta devono essere valutate una sola volta, usando il valore così calcolato anche per le altre occorrenze della stessa sottoespressione. Infine, un assegnamento è una particolare forma di espressione complessa. Si consideri il seguente frammento di codice:

```
int x = 2;
int A[5];
for (int i=0; i<5; i++)
   A[i] = i;
A[x++] = A[x++]+A[x++];
```

   Qual è lo stato del vettore `A` dopo l'assegnamento?

7. È dato il seguente frammento di codice in uno pseudolinguaggio con `goto`, scope dinamico e blocchi annidati etichettati (indicati con `A :{...}`):

```
A: { int x = 5;
     int y = 4;
     goto C;
        B: {int x = 4;
            int z = 3;
            goto E;
            }
        C: {int x = 3;
            int y = 10;
            D: {int y = 2;
                   }
            goto B;
            E: {int x = 1; // (**)
                int w = 1
               }
            }
     }
```

   Lo scope dinamico è gestito mediante tabella centrale dell'ambiente (CRT). Si illustri graficamente la situazione della CRT (esplicita, senza pila nascosta) nel momento in cui l'esecuzione raggiunge il punto segnato con il commento `(**)`.

8. Si dica cosa stampa il seguente codice Java. Se al posto della seconda e terza occorrenza di `this` scrivessimo `super` cambierebbe il risultato? Dare una breve motivazione delle risposte.

```
class A {
        int a=1;
        int f() { return -(this.a); }
        int g(int x){ return x*10;}
}

class B extends A {
        int a=2;
        int f() { return this.a; }
}

class C extends B {
        int a=3;
        int g(int x){ return x*30;}
        int f() { return g(this.a); }

}
public class pippo{
        public static void main(String args[]){
                C ccc= new C();
                B bbb =  ccc;
                System.out.println(bbb.f());
                System.out.println(bbb.a);
        }
}
```